Optimizing ecology-friendly drawing of plans of buildings by means of grammatical evolution

Arturo de Salabert, Alfonso Ortega, Manuel Alfonseca

Escuela Politécnica Superior – Universidad Autónoma de Madrid

ABSTRACT

We explore the application of grammatical evolution to the automatic generation of plans of building with constraints. A BNF is presented that guarantees the conversion of the genetic code into a well formed geometrical figure or phenotype. The validity of the approach is demonstrated, its limitations are analyzed and new evolutionary techniques are suggested for future work in this area.

1. INTRODUCTION

In the last decades, computer science has witnessed an interesting search for inspiration in Biology to help solve many different sorts of problems. Evolutionary computing, one of the results of this search, has created a number of constructs which have been applied successfully to multiple problems. Genetic programming (GP) has proved to be an effective and efficient generic search and optimization method in a wide variety of situations, while grammar evolution (GE), introduced by [ONeill 2001], has been elegantly applied to automatic programming [ONeill 2003] or to automatic music composition [Ortega 2002].

When applicable, GE has several advantages with respect to GP. From our perspective, the most relevant is the separation of the genetic code from the phenotype. This provides a great degree of freedom in developing or adapting the phenotype, without modifying the low level genetic processes. This is another example of the "onion principle", whose advantages are well known in fields like communications (ISO layers) or software engineering (OS levels).

Grammar-based drawing has been proposed [Ortega 2003] for the drawing of fractals by using a turtle like coding and L-grammars, but these studies did not need to address the two limiting problems which are encountered in our case: closure and the search space, as will be explained later.

Our approach starts by considering the plan of a building, not just as a group of lines, but as the result of a process of adapting to an environment certain user requirements, which impose a number of physical limitations. From the biological perspective, developing a building would be similar to growing a tree or a plant from a seed. The genetic code carries the characteristics of the specific variety of tree (the requirements for the building) while the place where the seed has fallen will set the limitations or constrains to its growth. The final objective of such an approach would be to automate the generation of plan drawings adapted to the environment, while, at the same time, fulfilling the requirements imposed by the designers. As a result of this process, we would have a variety of *seeds* that would compete between them to develop the best solution. Buildings of different shapes and characteristics would *grow* from these seeds, adapted to the particular circumstances, and the designer would have a set of choices to start the final draft, or even a set of solutions to choose from.

In this exercise we have chosen plans of buildings because it is at the same time a complex problem, rich in features and full of constraints, and also a familiar one, easy to explain. Machine drawing, for instance, would be another comparable problem, but not as intuitive.

2. GRAMMAR EVOLUTION

Grammatical Evolution [ONeill 200l] is a grammar based, linear genome system, which has been applied in the area of Automatic Programming to automatically generate programs or expressions in a given language that solve a particular problem. In Grammatical Evolution, the Backus Naur Form (BNF) of the grammar of the language is used to describe the output produced. Different BNF grammars can be used to automatically produce code in any language.

In Grammatical Evolution, the genotype usually is a string of 8 bit binary numbers generated at random, treated as integer values from 0 to 255. The phenotype may be a running computer program generated by a genotype-phenotype deterministic mapping process which translates the genotype, codon by codon from left to right, in the following way: the mapping begins with the axiom; at each step, the leftmost non terminal symbol of the current sentential form is chosen; the current codon is used to select the rule (among those available for that symbol) which is numbered with the current codon *mod* the number of rules for the current non terminal; a new sentential form is ob-

tained by applying the rule to the non terminal. When the string of integers in the genotype is exhausted before the phenotype has been completely generated, a biologically inspired wrapping mechanism is used to reuse the integers, similar to the gene-overlapping phenomenon observed in many organisms in nature. The mapping benefits from genetic code degeneracy, i.e. different integers in the genotype generate the same phenotype, according to Kimura's neutral theory [Kimura 1983].

In Grammatical Evolution, standard genetic algorithms are applied to the different genotypes in a population, using the typical crossover and mutation operators. For each domain, one must design the proper fitness function, which will be used by the genetic algorithm to perform selection.

3. PROBLEM DESCRIPTION

We suggest a *think big, start small* approach, and will start by the low hanging fruits, focusing first on a simple problem: how to draw the largest geometrical figure in a plot of land covered with trees. We will assume that local regulations penalize felling and limit the maximum size, while the minimum sizes are determined by practicality. Further constraints are easy to add: cost, orientation, shape... but our initial objective is not to make a realistic plan, but to experiment with the evolutionary approaches.

As our first simple case we'll assume a square plot of land of side n with some trees inside of side 1. The first objective of our evolutionary algorithm is to draw the largest figure with the minimum felling.

Of course, we could find very efficient geometrical solutions to this problem, but keep in mind that our objective is more ambitious. Our toy system will be implemented subject to the following considerations: A flexible case should not have a predefined shape; therefore the genetic code cannot have a limited length. There are several conditions to be fulfilled (which could change) and an obvious closure problem. Leaving all this to the fitness function is a significant overload. The phenotype correctness should be insured without affecting the genetic process. Complexity should be scalable without changing the basic genetic rules. Enable evolution and adaptation to the environment.

The GE system proposed here provides an answer to each of the previous points: Variable length genetic strings can be used, decoupled from the actual drawing.
The suggested Backus-Naur (BNF) grammar takes care of the fulfillment of any conditions as well as the closure problem. This is done without putting any extra load on the fitness process. The unconstrained search of genotypes is compatible with the production of a syntactically correct plan to be evaluated, as the genetic process (selection, crossing-over, mutation) does not operate on the actual plan, but on the strings that make up the genotype.
The procedure can be used to generate any type of drawing. Evolution and adaptability to the environment are enabled at the phenotype building time, by means of a two level process: coding the phenotype and expressing it within the constraints imposed by the environment.

In addition, genetic diversity and resilience to mutation are obtained by degeneration, and last but not least, it will be shown that a significant reduction of the search space is obtained.

4. SOLUTION DESCRIPTION

The proposed GE approach consists of the following components:

- A genetic algorithm based on the random creation of a first generation of individuals, which gives rise to their evolution based on selection, coupling and reproduction.
- A BNF grammar representing the *language* that will be used to generate a family of plans
- The phenotype developing process, which has been divided in two parts, coding of the potential phenotype and making it grow to its maximum extension.

The initial approach tested is basically equivalent to the procedure introduced in [ONeill 2001] with the addition of environment adaptation elements, which greatly enhance the effectiveness of the process.

The genetic code is formed by a sequence of 8-bit integer numbers (codons) which control the selection of the production rules by a simple transcription process (n *mod* m) where n is the integer that represents the codon (int 0..255) and m is the number of choices for the appropriate production rule. The result of this operation is mapped to the corresponding rule according to the matching number in square brackets. If for a particular rule there is only one choice, it is immediately expanded without consuming any element of the genotype.

4.1. A BNF for drawings

Any BNF grammar is a tuple {N,T,P,S}, where N is the set of non-terminals, T is the set of terminals, P is the set of production rules mapping N to {N∪T}*, and S is a member of N acting as the start symbol. In the proposed BNF for the coding of geometric figures:

```
N={<expr>,<orig>,<struc>,<form>,<shape1>,
  <shape2>,<angle>,<close>,<len>}
T={0,1,2,3,4,5,6,7,C,L,<,/,\,I,S,4',Ω,π,
   int}
S=<expr>
```

And the production rules P are:

```
(1)<expr>  ::= <orig><struc>
```

```
(2)<struc> ::= <angle><form><struc>     [0]
             | <close>                  [1]
(3)<form>  ::= <shape1><len><len>       [0]
             | <shape2><len><len><len>  [1]
             | C <angle><len>           [2]
(4)<orig>  ::= <len><len>
(5)<angle> ::= 0 | 1 | 2 |...| 7      [0-7]
(6)<shape1>::= L | < | \ | I           [0-3]
(7)<shape2>::= 4' | Ω | π              [0-2]
(8)<len>   ::= int ,, int ∈ [min,max]
(9)<close> ::= I | L | S | < | /       [0-3]
```

Where C represents a sector of *angle* degrees of an ellipse with eccentricity *len*; angle is a quantized value in increments of 45º; <, L, and \ represent two lines at 45º, 90º and 135º respectively; and 4', Ω and π are 3 parametric forms representing their corresponding shapes. This simple BNF makes it possible to build a large number of shapes. For example, the codings:

```
1. x0 y0 0 L d1 d2 I
2. x0 y0 2 C  6 d1 I
3. x0 y0 3 L d1 d2 L
4. x0 y0 1 \ d1 d2 7 L d1 d1 L /
5. x0 y0 4 L d1 d2 4 < d3 d4 I
6. x0 y0 4 L d1 d2 4 < d3 d4 L
7. x0 y0 4 L d1 d2 4 < d3 d4 S
```

where:

- d1, d2... represent different values each time.
- L closure (e.g. 6) implies a random choice of r.

Following a clockwise order, as shown in Fig 1, the examples generate the drawings show in Fig 2.

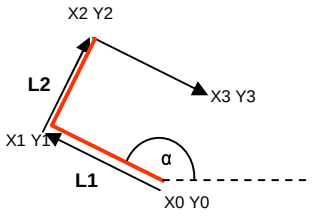

Fig 1

Notice that, while the translation from genes to drawing (phenotype) is deterministic, the same figure can be described by more than one code. For instance, example 4 can also be described by

```
x0 y0 1 \ d1 d2 \ d1 d1 L /
```

This is not a problem; on the contrary, it is another type of genetic degeneracy which increases the resilience of the genetic variety of the population.

For our initial experiments we have introduced further simplifications to reduce the search space. The simplified version of the BNF becomes:

```
(1)<expr>::=<len><len><angle><shape1>
           <len><len><close>
(2)<angle> ::= 0 | 1 | 2 |...| 7      [0-7]
(3)<shape1>::= L                        [0]
             | <                        [1]
(4)<len>   ::= int ,, int ∈ [min,max]
(5)<close> ::= I                        [0]
             | L                        [1]
```

Therefore the simplified rule set has the following choices:

Rule	Choices
1	1
2	8
3	2
4	N
5	2

In this case, our genotype is always 7-codon long and can be converted with the choices vector (0 0 8 2 0 0 2), where each integer correspond to the previous variable m used in the modulo operation, except when m=0, in which case the translation is based on

```
min + n·mod[max-min]
```

where `min` and `max` are the minimum and maximum values allowed for the length dimensions.

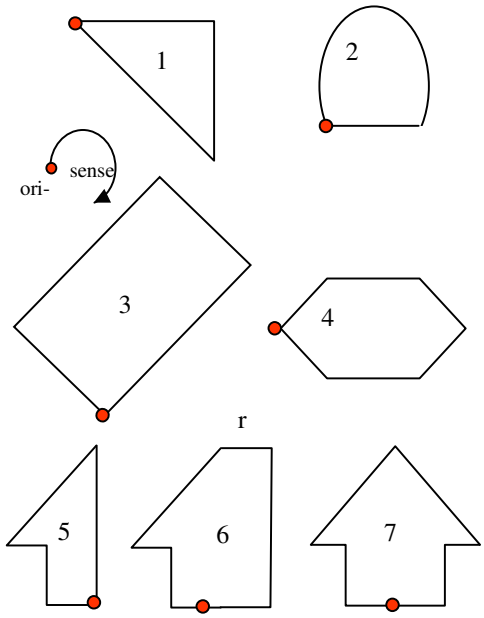

Fig 2

4.2. Reduction of search space table

In a square plot of side n, the search space grows with 2^N, where $N=n^2$. A quick estimation of the number of valid individuals (those allowed by the BNF) gives $3*n^3$, while the actual search space, using the simplified version of the

BNF, grows with $20.5*n^4$ if the maximum building surface is limited to 80% of the plot. As shown in table 1, although the introduction of a BNF has reduced enormously the search space, the likelihood of generating a valid individual is still very small, about 1.5% for a 10×10 plot. And we are talking of a toy example. At least a 50×50 plot should be used for any practical use.

n	Sq. size	Abs. Max.	$3*n^3$	BNF space	Viable indiv.
5	25	3,4E+07	375	12.800	2,9%
10	100	1,3E+30	3.000	204.800	1,5%
20	400	2,6E+120	24.000	3,3E+06	0,7%
30	900	8,5E+270	81.000	1,7E+07	0,5%
50	2.500	3,7E+753	375.000	1,3E+08	0,3%

Table 1. Search Spaces

The table shows that in GA a binary coding of length L produces a search space of size 2^L, while in GE this is reduced to $\Pi_{i=1,n}C_i$, where C_i is the number of choices for rule i. Choosing wisely a BNF can reduce enormously the search space. To get some feeling for these numbers, remember that a classical AI book [Winston 1992] compares 2^{400} to the number of chessboard configurations in a game of 100 moves, defining it as a *ridiculously large number*, even compared with the number of atoms in the universe (estimated to be less than 2^{300}).

However, there is still a huge room for improvements. The two main factors producing this still too large number are the start position and the two lengths of the L or < forms. We propose a number of solutions and will experiment with them shortly.

4.3. Fitness function rationale

The obvious fitness function for this problem computes the area of the plan defined by the genome, subtracting some penalty for felling trees. The question is what should be the penalty. Let's assume there is a single tree in a more or less central location, as in Fig 3.

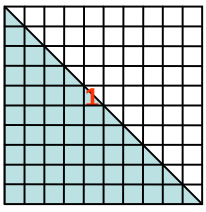

Fig 3

The fitness function has to make it unworthy to fell a single tree, otherwise the obvious solution of using the whole plot (or the maximum allowable surface) would be a better choice than avoiding felling the tree. This means that using about half the surface has to be a better choice than using the whole and felling the tree.

The penalty for felling cannot be a fixed value, because, as the surface increases, the penalty per tree felled would diminish. We came to a heuristic penalty of 2.5*n where n is the number of trees affected. To further reinforce a sharp fit to the maximum available surface, a tree fully inside the selected area counts as 1, but if it is on the wall it counts only as 0.5. This is justified by the fact that a tree on the wall can be spared at an acceptable cost (by using an omega shape around it, as in the complete BNF).

5. EXPERIMENTS AND RESULTS

We have run a number of trials with the simplified BNF for a 10×10 plot, using generations of 32 individuals and limiting the number of cycles to 200000 or 300000. This number is unacceptably large for a practical solution, but it is appropriate for our initial analysis. In the worst case, we could have generated 16 times each possible outcome from the BNF. In this case, it would be better to sequentially try all the possible individuals. Of course, this is not so, for crossover does not search the whole space, only the areas around the best individuals which chance created. But in most cases an acceptable solution was found much earlier. Mutation provides a broader exploration, but this genetic operator has a very small likelihood of creating a valid individual: 98.5% of the mutations are just thrown away. This also happens with the randomly generated first generation. It is quite remarkable that even with such small chances the system is able to find a good solution in a relatively small number of cycles.

In each generation, the best half of the population is kept, while the other half is replaced by new individuals, obtained from random coupling amongst the selected parents. To avoid the excess of reproduction of the fittest, which results from a roulette based coupling, we limit offspring to a couple of children for each couple of parents. Offspring from twin parents is also forbidden, forcing mutation when there is no other possible coupling available.

Results show what some authors have pointed out before: as long as there is genetic diversity, and under certain circumstances, the efficiency of crossover is much higher than that of mutation [Syswerda 1989], although there is no general agreement on that [Spears 2000]. In any case, once the genetic diversity has been lost, mutation is the only way out. This is, however, very time consuming.

Table 2 summarizes 26 runs of our experiment. The last two lines show the unweighted averages of the total set of runs and of the subset formed by runs 1-16. 62% of the runs found the best solution in less than 30.000 cycles (an average of 94 seconds in a 2005 desktop PC).

Even more interesting was to notice that a second best solution was found in 92% of the cases in less than 5.000 cycles (or 8 seconds). This is a noticeable result, compared with those found in the research for comparable search spaces, like in [Spears 2000] and others. This can be reduced enormously by improving the genotype-phenotype translation.

On the one side, it has been shown that only 1.5% of the generated individuals are valid (this is reduced to 0.7% in a 20×20 plot). In addition, we observe a very efficient convergence to very good results in a few cycles. We propose to modify the GE translation process to take the environment into account, either by reducing the too large individuals or by growing the small ones, which in both cases are discarded in the current approach.

Search for the best individual				Search for the 2nd best				
	Num. Gener.	t (s)	Gen/s		Num. Gener	t (s)	Gen/s	
1	3	0.03	100.0	1	2	0.02	100.0	
2	56	0.8	74.7	8	14	0.04	316.3	
3	75	1.0	75.0	2	16	0.2	74.7	
4	190	0.6	306.5	15	16	0.2	78.7	
5	1124	15.3	73.3	4	18	0.06	306.5	
6	1803	24.0	75.0	3	23	0.3	75.0	
7	2074	27.3	76.0	13	32	0.4	76.2	
8	3128	9.9	316.3	7	37	0.5	76.0	
9	3925	12.8	306.9	25	44	0.6	75.4	
10	5415	71.3	75.9	12	60	0.8	72.4	
11	13412	179.6	74.7	5	77	1.1	73.3	
12	14415	199.2	72.4	10	123	1.6	75.9	
13	15283	200.6	76.2	14	227	0.9	263.4	
14	15451	58.7	263.4	16	308	4.2	74.1	
15	24409	310.3	78.7	26	464	6.5	71.0	
16	29089	392.8	74.1	9	512	1.7	306.9	
17	4.143	117.9	348.9	20	526	6.8	76.9	
18	67153	239.2	280.7	21	706	9.5	74.3	
19	97723	1339.3	73.0	6	838	11.2	75.0	
20	125077	1626.5	76.9	23	1460	18.9	77.1	
21	200000	2692.6	74.3	17	2810	8.1	348.9	
22	200000	2656.1	75.3	11	3455	46.3	74.7	
23	200000	2594.3	77.1	24	4323	59.0	73.3	
24	200000	2730.3	73.3	18	4337	15.4	280.7	
25	232954	3091.5	75.4	22	5297	70.3	75.3	
26	300000	4222.6	71.0	19	8234	112.8	73.0	
	68996	**877.5**	**128.6**		**1306**	**1.45**	**128.6**	**T**
62%	**8116**	**94.0**	**132.4**	**92%**	**851**	**8**	**133**	**S**

Table 2 (T: total averages, S: subset averages)

6. CONCLUSIONS & FUTURE WORK

The problem of drawing the plan of a building subject to requirements and constraints has been solved using Grammatical Evolution. The system finds good solutions quickly, but it may take much longer to find the best possible one.

Crossover seems to be a much more efficient genetic operator procedure than mutation, as long as there is sufficient genetic diversity. As a corollary, it is very important to have a good initial population.

Blind and random search could be improved by combining this technique with others which exploit the adaptation to the environment.

At least three possible lines of improvement have been identified and will be explored.

7. REFERENCES

[Kimura 1983] M. Kimura: *The neutral theory of molecular evolution*. Cambridge University Press (1983).

[ONei11 2001] M. O'Neill and C. Ryan, "Grammatical Evolution," *IEEE Trans. Evolutionary Computation* 5, No. 4, 349–358 (2001).

[ONeill 2003] Michael O'Neill and Conor Ryan, "Grammatical Evolution: Evolutionary Automatic Programming in an Arbitrary Language," Kluwer Academic Publishers, 2003, 160 pp; *Book Series: Genetic Programming*: Volume 4, ISBN 1-4020-7444-1.

[Ortega 2003] A.Ortega, A.Abu Dalhoum & M.Alfonseca, "Grammatical evolution to design fractal curves with a given dimension," *IBM Journal of Res. and Dev.*, Vol. 47:4, p. 483-493, Jul. 2003.

[Ortega 2002] A.Ortega, R.Sanchez Alfonso & M.Alfonseca, "Automatic Composition of Music by means of Grammatical Evolution," *APL Quote Quad (ACM SIGAPL)*, Vol. 32:4, p. 148-155, Jun. 2002.

[Syswerda 1989] G. Syswerda. "Uniform crossover in genetic algorithms." *Proceedings of the 3rd International Conference on Genetic Algorithms*, pages 2–9, 1989.

[Spears 2000] William M. Spears. "Crossover or Mutation?" *Foundations of Genetic Algorithms Workshop (FoGa) 2000*. Charlottesville, VA

[Winston 1992] Patrick H. Winston. *Artificial Intelligence (3rd Edition)*. Addison-Wesley, 1992